التعامل مع المؤشرات Pointers في لغة C++

1. مواقع الذاكرة

يمكن أن نعتبر شكل الذاكرة مثل صناديق البريد ، حيث تمثل كل خانة موقع في الذاكرة و تقوم بتخزين قيمة وحيدة (قد تكون صحيحة int او حقيقة float او char) ، كل موقع في الذاكرة له عنوان ، و يتم تمثيل عناوين الذاكرة باستخدام النظام الست عشري من باب التسهيل كون انه في الاصل يمثل باستخدام النظام الثنائي ،فعلى سبيل المثال لتمثيل الخانة رقم 15 باستخدام النظام الثنائي فإننا سوف نحتاج إلى اربع خانات هي 1111 على عكس النظام الست عشري الذي يمثلها بخانة واحدة هي F .

2. حجم الأنماط في الذاكرة

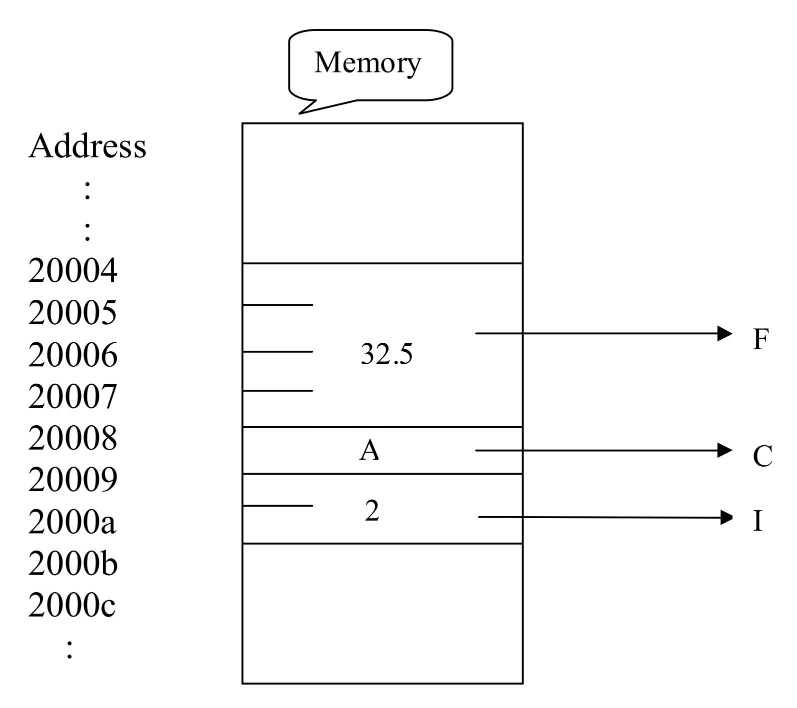
قبل أن نخوض بالمؤشرات لا بد من التذكير بطريقة عنونة المتحولات بالذاكرة .
عندما نقوم بتعريف المتحولات في البرنامج ، فإنها تأخذ حيزاً في الذاكرة بما يتناسب مع حجمها

1	**Char**
2	**Int**
4	**long**
4	**float**
8	**double**

فعلى سبيل المثال :

```
void main()
{
 float  F=32.5;.
 char  C='A';
  int    I= 2;
}
```

فسيتم حجز 4 حجرات للمتحول F وحجرة للمتحول C وحجرتين للمتحول I وسيكون تمثيلهم بالذاكرة على الشكل التالي



3. الحجز الاستاتيكي

فكما رأينا فإنه يتم حجز حجم المتحولات تبعاً لنمطها و هذا الحجز يسمى بالحجز الساكن الستاتيكي (static variable) ويوصف بالساكن لأن الذاكرة المكرسة له تبقى محجوزة له طوال مدة تنفيذ البرنامج, فالمترجم هنا يعرف تماما كمية الذاكرة التي تحتاجها المتحولات , ويخصص المترجم حجرات المتحولات العامة والثوابت ضمن مقطع المعطيات (data segment) ومقطع المعطيات هذا عبارة عن مساحة من الذاكرة محدودة الطول , يحدد ويقرر حجمها المترجم بناء على عدد ونوع المتحولات العامة والثوابت المصرح عنها بالبرنامج , أما المتحولات المحلية (Local Variable) والبارمترات فالمترجم يخصص لها حجرات في الذاكرة عندما يتطلب تنفيذ البرنامج ذلك , ولكن يحجز المترجم بشكل أولي كمية من الذاكرة المكدس (Stack)من أجل هذه المتحولات –وطول مقطع المكدس ثابت ويحدد أثناء الترجمة للبرنامج –
فالخلاصة أن الحجز الستاتيكي يتم تحديد كمية الذاكرة المخصصة أثناء ترجمة البرنامج وقبل تنفيذه, لأن هذه الكمية ثابتة لا تتغير .

4. الحجز الديناميكي

لنفرض أننا نريد برنامج من أجل ترتيب معطيات عددية يدخلها المستخدم و لا أعرف هذه الكمية والكمية ليست ثابته فمن أجل ذلك يوجد طريقتين
الأولى : أن أحجز مصفوفة ذات حجم كبير بحيث تكفي لتخزين القيم المتوقعة و هنا سنحجز كمية كبيرة من الذاكرة عن طريق الحجز الاستاتيكي
الثانية : عن طريق استخدام المؤشرات والحجز الديناميكي
إن المبدأ الاساسي للتعامل مع المتحولات الديناميكية هوتخصيص ذاكرة ديناميكية للبرنامج يتم الحجز ضمنها لهذه المتحولات ولاستخدامها وبعد الانتهاء يتم تحرير الحجز (إعادة الذاكرة المحجوزة لاستخدامات اخرى) فيما بعد.
إذا نحن الان سنتعامل مع طريقة جديدة مع الذاكرة و هي تخصيص حجرات الذاكرة أثناء التنفيذ (تنفيذ البرنامج) وليس أثناء الترجمة (compile) ترجمة البرنامج .
ولكن كيف يمكننا التعامل هذه المتحولات الديناميكية وكيف سوف نصرح عنها ؟

تتم طريقة الحجز باستخدام مؤشرات (عناوين) إلى الذاكرة -الديناميكية – والتي تسمى الكومة Heap وستتم بهذه الطريقة بسبب فكرة الاخذ والاعادة منها وإليها , والخلاصة أن استخدام المتحولات الديناميكية لا يتم مباشرة بل عن طريق مؤشر.

5. عناوين المتحولات في الذاكرة

لكل خلية من خلايا ذاكرة الحاسب عنوان محدّد, والعنوان هو رقم يبدأ من 0 وينتهي بالقيمة العظمى لسعة الذاكرة.
كي نستطيع ان نعرف عنوان متحول في الذاكرة نقوم بإضافة الإشارة & قبل المتحول و هي تعني عنوان المتحول (Address Of) ،لاحظ :

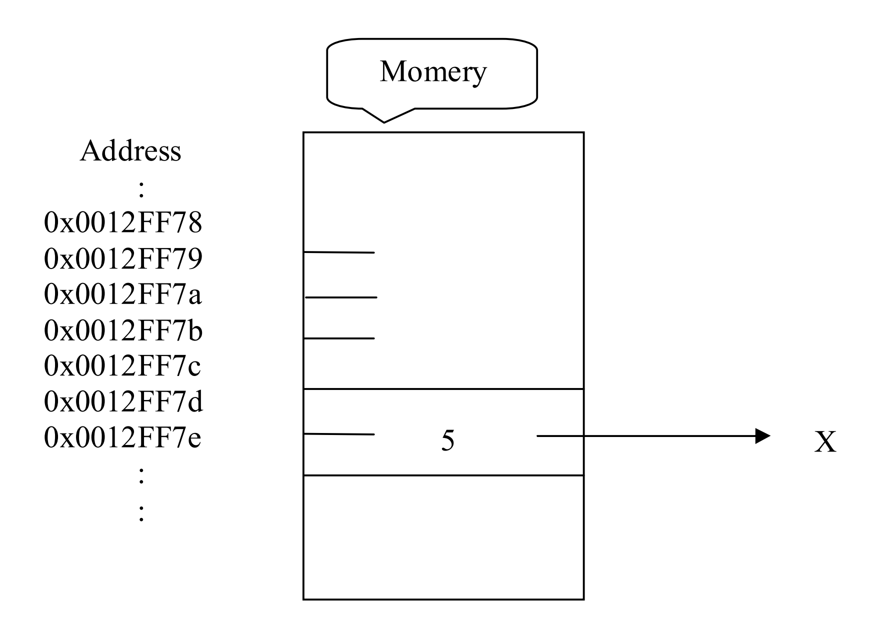
```
#include "iostream.h"
#include "conio.h"
void main()
{
        int x;
        cout << "Enter Number: ";
        cin   >> x;
        cout << "\nThe Number is:" << x << endl;
        cout << "\nand The address in memory is: "<< &x << endl;
        getch();
}
```

يظهر محتوى المتحول

يظهر عنوان المتحول

عند تجربة البرنامج تظهري لي النتيجة التالية :

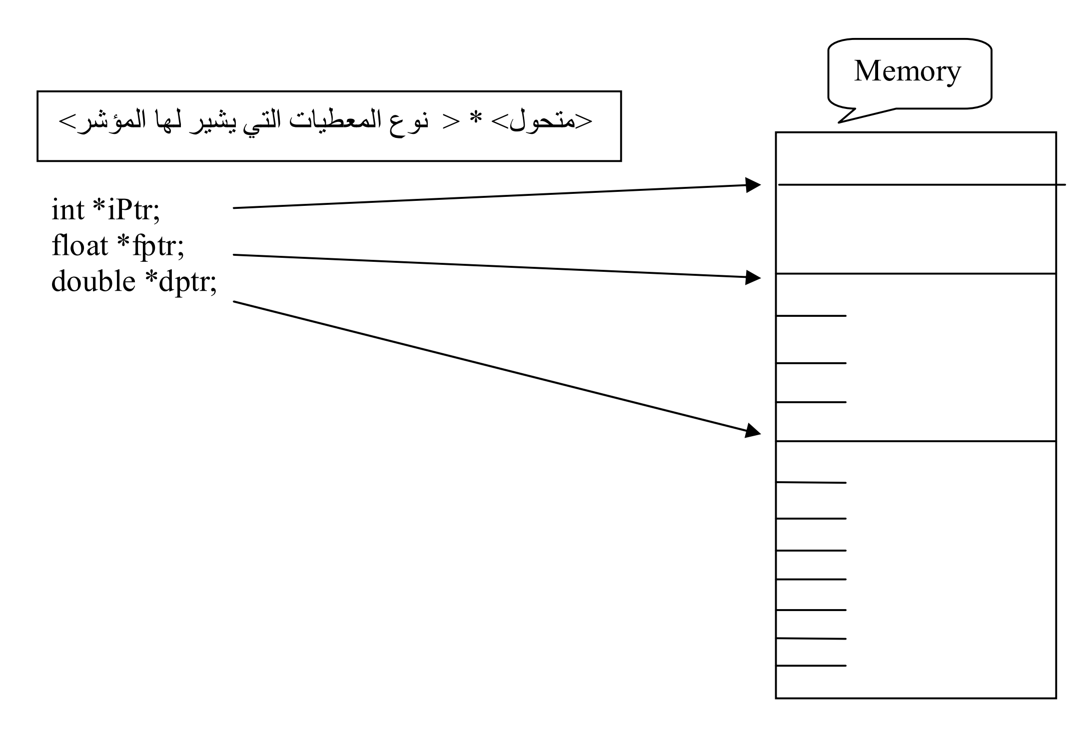
```
"C:\DOCUMENTS AND SETTINGS\ADMINISTRATOR\DESKTOP\EMWINTRIAL_V396_1BPP\SAMPL...
Enter Number: 5
The Number is:5
and The address in memory is: 0x0012FF7C
```



6. المؤشرات والتصريح عنها

المؤشر عبارة عن متحول يحتوي على عنوان في الذاكرة (و ليس قيمة عادية) ، و هو يشير الى (Point to) العنوان الذي يحتويه و بالتالي فهو يشير إلى متحول آخر ...

يتم التصريح عن المؤشر في البرنامج بتحديد نوع المعطيات التي يشير اليها (أي هل يشير إلى قيمة من نوع int-char-float ..) ثم اضافة العلامة (الرمز) نجمة * ثم اسم المؤشر.



لا حظوا معي:
المتحول iptr هو عبارة عن مؤشر من النوع الصحيح , أي عنوان في الذاكرة يتألف من حجرتين ومحتويات هاتين الحجرتين سوف تفسر على أنها أعداد صحيحة .
المتحول fptr هو عبارة عن مؤشر من النوع الحقيقي , أي أن العنوان المخزن في المتحول يفهم على أنه عنوان بداية مساحة من الذاكرة بطول أربعة بايتات ومحتويات هذه الحجرات أعداد صحيحة .
المتحول dptr هو عبارة عن مؤشر من النوع المضاعف , أي أن العنوان المخزن في المتحول يفهم على أنه عنوان بداية مساحة من الذاكرة بطول ثمانية بايتات ومحتويات هذه الحجرات أعداد من النمط المضاعف .

تفقنا ان الرمز & يعيد عنوان المتغير في الذاكرة ، لاحظ هذا المثال :

```
int y = 5;
int *yPtr;
yPtr = &y; // y تأخذ عنوان yPtr
```

ان انشاء المؤشر يتم على مرحلتين:
الاولى نعلن فيها عن المؤشر yPtr و الثانية نستد اليه عنوان متحول في الذاكرة و ذلك يعني ان yPtr سوف يشيرإلى المتحول y في الذاكرة و بالتالي من الممكن التعامل مع y بشكل غير مباشر عن طريق **yPtr** أي بمعنى أخر أصبحت القيمة داخل y الخمسة هي نفسها محتوى القيمة التي يشر اليها المؤشرyPtr .
كل المتحولات من نوع مؤشر لها نفس الحجم في الذاكرة و هو حجم العنوان الذي تحتويه (فهي مجهزة لتخزين العناوين)

Example:

```
// PoiterAddress.cpp :
#include <iostream.h>

int main()
{
        int x = 1, y = 5;
        cout<<endl<<&x<<endl<<&y;
        int * ptr;
        ptr = &x;
        cout<<endl<<ptr;
        ptr = &y;
        cout<<endl<<ptr;
        return 0;
}
```

```
x0065FDF40
0x0065FDF0
0x0065FDF4
0x0065FDF0
```

إن هذا البرنامج يقوم بتعريف المتحولين x , y ويهيئهما بالقيمتين 5 , 1 ومن ثم يقوم بطباعة عناوينهما, ومن ثم يقوم بتعريف متحول من نوع مؤشر إلى قيمة صحيحة وذلك من خلال الأمر: (int * x). من المعلوم أنه لدى تعريف متحول ما فإنه لن تكون له أية قيمة, وبالتالي يمكن أن يأخذ قيمة عشوائية, ولكن في المؤشرات فإن هذه القيمة تمثل عنواناً من الذاكرة. ولذا يتوجب قبل استخدام أي مؤشر أن نضع فيه عنواناً محدّداً.
وخلاصة القول : إن المؤشر يحمل عنوان متحول من نوع ما, وهذا العنوان يجب تحديده وإلا فإن المؤشر قد يشير إلى غير المكان المقصود.

7. غاية المؤشر

عندما نعرف عن مؤشر فإننا نستخدم الصيغة (int *varPtr) حيث ان اسم المؤشر هو varPtr ، و لكننا نستخدم الصيغة varPtr*داخل البرنامج (كتعبير) من اجل الوصول إلى محتويات المتحول الذي يشير اليه المؤشر varPtr ، ويسمى غاية المؤشر

```
void main ()
{
int *varPtr;
Int var = 20 ;
varPtr = &var;
cout << *varPtr; // 20
cout << var ; // 20
}
```

يستخدم اسم المؤشر مسبوق بالنجمة في البرنامج ، من اجل الوصل إلى محتويات المتحول الذي يشير اليه,ويسمى غاية المؤشر



و هكذا علينا الانتباه إلى :

- استخدام varPtr* في جملة الاعلان int *varPtr; تعني اننا نعلن عن مؤشر اسمه varPtr و يشير إلى قيمة من نوع عدد صحيح .
- استخدام varPtr* في البرنامج كتعبير ;cout << *varPtr تعني اننا نتعامل مع محتوى القيمة التي يشير اليها المؤشر varPtr .

8. الوصول إلى المتحول المشار إليه :

يمكن الوصول إلى محتويات متحول لا نعرف اسمه ولكننا نعرف عنوانه كما في المثال التالي:

Example:

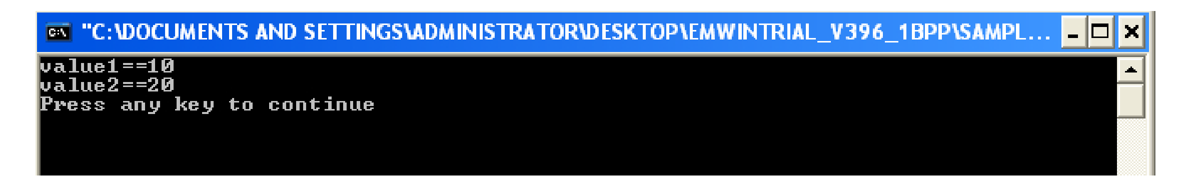
```
// PointersSource.cpp :
#include "stdafx.h"
#include <iostream.h>

int main()
{
        int x = 1, y = 5;
        int * ptr;
        ptr = &x;
        cout<<endl<<*ptr;
        ptr = &y;
        cout<<endl<<*ptr;
        return 0;
}
```

عند وضع عنوان متحول ما ضمن مؤشر يجب أن يكون المتحول والمؤشر من نفس النوع و لا يمكن وضع عنوان متحول من نوع float ضمن مؤشر إلى النوع int.
لكن تعريف المؤشر مؤشر إلى void مثل (void * ptr) يجعل من الممكن لهذا المؤشر أن يشير إلى أي نوع من المعطيات

Example:

```
int main ()
{
        int value1 = 5, value2 = 15;   // الاعلان عن متحولين صحيحين
        int* p1;                       // الاعلان عن مؤشر يشير إلى قيمة من نوع عدد صحيح
        int* p2;                       // الاعلان عن مؤشر ثاني يشير إلى قيمة من نوع عدد صحيح
        p1 = &value1;                  // جعل المؤشر الاول يشير إلى المتحول  الاول
        p2 = &value2;                  // جعل المؤشر الثاني يشير إلى المتحول الثاني
        *p1 = 10;                      // تخزين القيمة 10 في المتحول الذي يشير اليه المؤشر
        *p2 = *p1;                     // مساواة قيم المتحولات التي تشير لها المؤشرات
        p1 = p2;                       // جعل المؤشر الثاني يشير إلى المتحول الذي يشير اليه المؤشر الاول
        *p1 = 20;                      // تخزين القيمة 20 في المتحول الذي يشير اليه المؤشر الاول
        cout << "value1==" <<value1<<" value2=="<< value2;
        return 0;
}
```


```
"C:\DOCUMENTS AND SETTINGS\ADMINISTRATOR\DESKTOP\EMWINTRIAL_V396_1BPP\SAMPL...
value1==10
value2==20
Press any key to continue
```

ملاحظات:

يمكن للمؤشرات أن تأخذ عنواناً جديداً بالمعامل new أو أن تأخذ عنوان أحد المتحولات التي تنسجم مع ما تؤشر عليه (أي لا يجوز إسناد عنوان متحول int إلى موشر على float) والحجز من خلال المعامل new يتم بالشكل التالي:

PointerVar=new datatype;

حيث datatype هي نمط قيمة المؤشر عندما صرحنا عنه و هنا يتم حجز مكان جديد في الذاكرة.
لا يمكن الوصول إلى محتوى المتحول الذي من نمط مؤشر إلا بعد أن تتم تهيئته (أي بعد أن يحمل عنواناً معيناً).
الحذف أي تحرير المنطقة الذاكرية المحجوزة لهذا المؤشر يتم من خلال التابع delete وفق الصيغة:

delete PointerVar;

القيمة الخاصة NULL تعني أن المؤشر لا يؤشر على أي قيمة و هي مختلفة عن المعامل delete فعند عملية التحرير هذا يعني أن المؤشر لم يعد له أي مكان في الذاكرة أي أن داخله لا يوجد عنوان ذاكري لحجرة ما بينما عند إسناد القيمة NULL هذا يعني أن له قيمة خاصة لا تمثل عنواناً فعلياً (مثل الصفر) والوصول إلى القيمة المحتواة ضمنه سيولد خطأ.
يجب الانتباه أن إسناد المؤشرات يؤدي إلى أن تحمل نفس العنوان و بالتالي أي تغيير في المحتوى يسري مفعوله على كلا المتحولين أما تغيير القيم فيأخذ مفعوله مرة واحدة و تبقى العناوين منفصلة علماً أن إسناد مؤشرات من قيم مختلفة

غير مقبول حتى لو كانت أكبر أي مؤشر على قيمة float لا يمكنه أن يحتوي على مؤشر على قيمة int بينما القيم التي هي من نمط int وfloat تخضع لنفس القواعد التي تعلمناها.

القسم الأول
المهندس : محمد ناشد